Intro to yal

João M. P. Cardoso
Email: jmpc@acm.org

yal

- Imperative programming language
- Programs organized as modules (one module per file)

yal

- Semantic of passing arguments and returning results equivalent to respective Java semantics
- All scalar variables are of type integer (corresponding to Java *int*)
- Arrays are unidimensional and an array variable represents a reference to the array
- Variables need to be initialized before used (size of the arrays needs to be defined)
- For unspecified semantics follow the respective Java semantics

Hello World!

```
module first_program {
    function main() {
        io.print ("Hello World");
    }
}
```

Particularities 1

```
module example {

   a[] = [10];    // the same as a=[10]; (only possible in attributes)
   a[] = 1;       // the same as a = 1; (only possible in attributes)

   N = 100;
   b[] = [N];  // N must be a constant and must be
               // defined before this statement
   b[] = 1;    // assigns 1 to all the elements of b
   b[0] = 100;  // assigns 100 to element 0 of b
...
```

Particularities 2

```
module example {

  function f() {
    a=[100];
    size1 = a.size;
    io.println("number of elements of a: ", size1);

    a.size = 20; // what is the semantic of this?

    b = [10];
    cmp = a < b; // what is the semantic of this?
                     // shall we interpret it as cmp = a.size < b.size;?
  }
...
```

Semantic error

Semantic error

Particularities 3

```
module example {

   function f3() {
      io.println(“aqui”);
   }
   function f1() {
       f2();
       f3();
   }
   function f2() {
      io.println(“aqui”);
   }
...
```

No semantic error!

Particularities 4

```
module example {
  function c=f3(a) {
      if(a==1) {
          b=[20];
      } else {
          b=2;
      }
      c = b;
  }
...
```

Semantic error: “b” may reach this position as a scalar or as an array

Particularities 5

```
module example {
  function c=f3(a) {
       if(a==1) {
           c=3;
       } else {
           b=2;
       }
       c = b;
  }
...
```

Semantic error: “b” may reach this position without initialization

Particularities 6

```
module example {
  function f3() {
       a=ext.f1();
       ext.f2(a);
  }
...
```

Is “a” a scalar variable or an array variable?
Undecidable without knowing the prototype of “f1” or “f2”.
By default we will interpret it as a scalar variable.

Particularities 6

```
module example {
  function f3() {
        a=ext.f1();
        ext.f2(a);
  }
...
```

Is “a” a scalar variable or an array variable?
Undecidable without knowing the prototype of “f1” or “f2”.
By default we will interpret it as a scalar variable.

Particularities 7

```
module example {
  function f3() {
        a=[0];
        a=ext.f1();
        ext.f2(a);
  }
...
```

To drive the compiler to consider “a” as an array variable we can define “a” as an array variable (possibly with size “0” and without generating code for this statement).

Example 1

```
module programa1 {

   data=[100]; // vector of 100 integers
   mx; // attribute mx
   mn; // attribute mn

   function det(d[]) {
      i=0;
      M=d.size-1; // d.size equivales to d.length (Java)

      while(i<M) { // version not optimized!
         a=d[i];
         i=i+1;
         b=d[i];
         mx= library1.max(a,b);
         mn= library1.min(a,b);
      }
   }

   function main() {
      det(data);
      io.println(“max: “,mx);
      io.println(“min: “,mn);
   }
}
```

Example 1

```
module library1 {

    function m=max(a,b) {
        if(a > b) {
             m = a;
        } else {
              m = b;
        }
    }

    function m=min(a,b) {
        if(a > b) {
             m = b;
        } else {
             m = a;
        }
    }

}
```

Example 2

```
module example {

function a[]=f1(b[]) {
  i = 0;
  a = [b.size];
  while(i<b.size) {
    a[i] = b[i];
    i = i +1;
  }
}
...
```

This function receives an array as parameter and returns an array

Creates an array of b.size elements of type int

Example 2

```
...

function a[]=f2(N) {
  a = [N]; // creates an array “a” with N integers
  a = 1; // initialize all the N elements of array
  “a” with value 1
}
...
```

yal and Java

```
module programa1 {

  data=[100]; // vector of 100 integers
  mx; // attribute mx
  mn; // attribute mn
...
```

yal

```
public class programa1 {

  public static int[] data = new int[100];

  public static int mx;
  public static int mn;
...
```

Java

yal and Java

```
module example {

function a[]=f2(N) {
  a = [N]; // creates an array “a” with N integers
  a = 1; // initialize all the N elements of array “a” with value 1
}
...
```

yal

```
public class example {

  public static int[] f2(int N) {
    int[] a = new int[N];
    for(int i=0; i<N; i++)
      a[i] = 1;
    return a;
  }
...
```

Java

yal and Java

```
module example {

   b;

   function set(N) {
      b = N;
   }

   function m=get() {
      m=b;
   }
...
```

yal

```
public class example {
   public static int b;

   public static void set(int N) {
      b = N;
   }

  public static int get() {
        int m;
        m = b;
        return m;
   }

...
```

Java

yal and Java

```
module example {


  b = [10];
  b = 1;


...
```

yal

```
public class example {
  public static int[] b;

  public static example() {
    b = new int[10];
    for(int i=0; i<b.length; i++)
      b[i] = 1;
  }
...
```

Java

yal and Java

```
module example {

   m;

   function m=f1(m) {
     // in this function we will
     // never have the possibility
     // to access the attribute m
           m=m+3;
   }

...
```

yal

```
public class example {
   public static int m;

   public static f1(int m)
  {
      m = m+3;
     return m;
  }
...
```

Java